VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Bakalářská práce

Rok 2010 Zdeněk Parma

VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra Informatiky

Jádro klienta XMPP s podporou multimédií
The Core XMPP Client with Support for Multimedia

Rok 2010

Zdeněk Parma

VSB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Zdeněk Parma**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Jádro klienta XmPP s podporou multimédií
The Core XMPP Client with Support for Multimedia

Zásady pro vypracování:

Navrhněte a pokuste se realizovat jádro klienta protokolu XmPP pro server vyvíjený v rámci práce Zbyňka Složila (SLO805). Klient bude mít kromě standardních funkcí možnost zasílání multimediálních dat.

Práce bude obsahovat:
1. Popište protokol XmPP.
2. Popište strategii, která bude použita při přenosu multimediálních dat.
3. Ve vámi zvoleném programovacím jazyku se pokuste realizovat navrhnuté funkce.
4. Výstupen bude program nebo programy, na kterých bude možné otestovat funkčnost všech částí systému.
5. Dbejte na dobrou dokumentaci.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Seidl**

Datum zadání: 20.11.2009

Datum odevzdání: 07.05.2010

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr.Ing. Eduard Sojka
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: Zdeněk Parma

Rád bych poděkoval Ing. Davidu Seidlovi za jeho cenné rady a trpělivost. Také bych rád poděkoval kolegům Tomáši Sikorovi a Zbyňku Složilovi za jejich spolupráci.

Abstrakt

Práce má za cíl seznámit s protokolem XMPP. Vytvořit jádro klienta pracujícího s tímto protokolem, schopného se připojit k libovolnému veřejnému XMPP serveru Jádro klienta neobsahuje, žádné uživatelské rozhraní a musí být připraveno na jednoduchou implementaci do uživatelského rozhraní.

Další částí této práce je vytvořit strategii pro přenos multimediálních dat.

Klíčová slova

XMPP, Jabber, Gloox, IM klient, XML

Abstract

This work has a goal to introduce with protocol XMPP. To create client core working with this protocol, able to connect to any public XMPP server. Client core don't contains any GUI and it must be ready for simple implementation of GUI.

Next part of this work is to create strategy for stream of multimedia data.

Keywords

XMPP, Jabber, Gloox, IM client, XML

Seznam použitých symbolů a zkratek

XMPP (Extensible Messaging and Presence Protocol)
IM (instant messenging) instantní komunikace, okamžité přeposílání zpráv v konverzaci.
XML (Extensible Markup Language), jazyk pro strukturované dokumenty.
XMPP Core – standard pro spojení v síti XMPP.
XMPP IM – standard pro výměnu zpráv v síti XMPP.
RTP (Real-Time Protokol) – protokol popisující výměnu multimediálních dat.
RTCP (Real-time Transport Control protocol) – slouží k řízení dat posílaných přes protokol RTP.
SDP (Session Description Protocol) – popisuje formát parametrů, které se vyměňují při přenosu multimediálních dat.
GUI (Grafic Usert Interface) - grafické rozhraní sloužící ke zobrazení výstupu, pro uživatele v přehledné formě
QT – knihovna pro tvorbu GUI.
SGML (Standard Generalized Markup Language) – značkovací jazyk.
W3C (The World Wide Web Consortium) – organizace spravující mezinárodní standardy pro www stránky.
IETF (Internet Engineering Task Force) – Skupina spravující standardy v síti Internet.
HTTP (Hyper Transport Protocol) – protokol používaný pro webové stránky.
VoIP (Voice over Internet Protocol) – protokol pro audio komunikaci.
Open source – otevřený, zcela dostupný kód.
IP – adresa v síti Internet.
DNS (Domain Name systém) – překládá jména domén na IP adresy.
TCP (Transmission Control Protocol) – Spojení s potvrzováním doručení.
UDP (*User Datagram Protocol*) - spojení bez potvrzování doručení.
QoS (Quality of Service) – určuje kvalitu proudu dat (například u videa, určuje kvalitu proudu dat pro plynulé přehrávání videa).
MUC (multi-user chat rooms) – místnost, kde může komunikovat více klientů najednou.
Javadoc – dokumentace generovaná z kódu jazyka JAVA
TLS (Transport Layer Security) – šifrování zabraňující odposlech na síti.
SASL (Simple Authentication and Secure Layer) – metody pro ověřování
Socket – spojení do sítě.
Parser – program na kontrolu a rozložení strukturovaného dokumentu.

SAX (Simple API for XML) – XML parser, který zpracovává dokument postupně.

DOM (Document Object Model) – XML parser, který načte celý dokument do paměti.

DTD (document type definition) – definuje značky ve strukturovaném dokumentu.

značka (tag) – představuje název, kterým se identifikuje element v XML.

Element – reprezentuje informaci ve strukturovan0m dokumentu.

Jabber – původní název protokolu XMPP.

Heart-Beat – funkce, která ověřuje zdali trvá spojení.

WLM (Windows Live Messenger) – IM klient od společnosti Microsoft.

ICQ (i seek you) – velmi rozšířený IM klient v České republice.

Service Discovery – zjišťuje dovednosti serveru nebo klienta v síti XMPP.

Disco – Service Discovery.

BOSH (Bidirectional-streams over synchronous http) [XEP-0124] - imuluje dlouhodobé TCP spojení mezi dvěma entitami prostřednictvím dotazů http.

Presence – XMPP strofa, udává stav kontaktu.

JID (Jabber ID) – identifikace entity v síti XMPP.

nahé JID (bare JID) – JID bez zdroje, určuje identitu, ale ne místo odkud se entita připojila.

plné JID (full JID) – JID se zdrojem.

Roster - seznam kontaktů v síti XMPP.

MINGW – kompilátor pro operační systém Windows.

Obsah

1. Úvod

1.1 Předmluva

Postupem času jsou čím dál větší nároky na rychlou a bezproblémovou komunikaci. Vynecháme-li prvopočátky komunikace pomocí poslů, poštovní holubů atd. a začneme od komunikace elektronické. Na začátku tu byl email, který úspěšně přetrvává doteď, jako spolehlivý, zaběhnutý a rozšířený standard. Pokud ovšem chceme komunikaci v reálném čase s co možná nejrychlejší odpovědí, je nutné hledat jinde. Dnes existuje celá řada protokolů pro instantní komunikaci, většina z nich je komerční. Takže jsou uzavřené a jejich používání je omezeno licenčními podmínkami. Bylo by skvělé, kdyby se v této instantní komunikaci objevil protokol, který by se snažil fungovat podobně jako email. Byl by to standard, byl decentralizovaný a mohl jeho server provozovat kdokoliv. Takový protokol existuje a nazývá se Extensible Messaging and Presence Protocol (XMPP[1]). Tento protokol se rozšiřoval jako im[2] síť velmi nenápadně. I když jej implementoval například google pro komunikaci přes google talk, v podstatě tajil že jde o XMPP. Poslední velký rozmach pro XMPP byl, když v současnosti největší světová sociální síť facebook[3] otevřela svůj integrovaný chat založený právě na XMPP. Znovu se však neví, že jde o XMPP. To však není důležité. To co je důležité je, že se XMPP rozmáhá a začíná tak tvořit opravdový standard v komunikaci..

Tento protokol se ovšem nehodí pouze pro komunikaci běžných uživatelů, pro níž byl původně navržen. Pro jeho velkou modifikovatelnost, lze použít v mnoha dalších odvětvích. Komunikují přes něj nejen lidé, ale i stroje, které si přes něj vyměňují informace o spojení apod.

XMPP je již dnes natolik rozšířené, že je více méně jasné , že tu s námi tento standard ještě nějakou dobu i vydrží. Proto jsem si zvolil tuto bakalářskou práci, abych pronikl do tajů tohoto protokolu, naučil se jak funguje, jak se s ním pracuje a co už kolem něj existuje.

1 XMPP (Extensible Messaging and Presence Protocol)
2 IM (instant messenging) – instantní komunikace.
3 Facebook – Velmi rozšířená sociální služba.

1.2 Rozdělení projektu

Problematika XMPP sítě zahrnuje server, klienta a uživatelské prostředí pro klienta. Protože to je na jednu práci příliš mnoho práce rozdělila se mezi tři studenty. Tito studenti jsou:

- Zbyněk Složil, ten vytváří jednoduchý XMPP server.
- Zdeněk Parma (autor této práce), který má na starost jádro XMPP klienta, přeposílání zpráv a multimediální spojení.
- Tomáš Sikora, který zpracovává grafické a konzolové uživatelské rozhraní pro XMPP klienta.

Protože se tato práce kříží s prací kolegů, jsou v tomto úvodu uvedeny citace, které krátce popisují práci kolegů.

Zbyněk Složil – XMPP server [1]

Moje práce v projektu je implementace XMPP serveru, jehož cílem je poskytnutí základních komunikačních dat klientovi a přeposílání zpráv mezi nimi po síti. Jako rozšíření by měla být multimediální komunikace. Server byl programován především pro práci s klientem od mých kolegů. Při implementaci nebyla použita skoro žádná pomocná knihovna pro usnadnění práce, a proto musel být server vytvořen od úplných základů, to jak část komunikace, tak i řešení samotných XML1 dokumentů. Zatím chybí odesílání chybových zpráv zpět ke klientovi, což i když patří k důležité části protokolu, neomezuje to funkčnost. Za to server poskytuje autentizaci klienta, stažení seznamu kontaktů, presenci a odesílání samotných zpráv. Je to teprve první verze mé implementace, která se stala kostrou většího projektu.

Práce navazuje těsně s prací Zdeňka Parmy a jeho jádrem klienta XMPP. Mezi námi jsou přeposílány XML zprávy, které můj server rozkóduje a přepošle na požadované místo. Tím je komunikace mezi námi přesně daná a nevznikají žádné rozpory.

Zdeněk Parma (autor této práce) – jádro XMPP klienta.

Jádro XMPP klienta tvoří spojení se s libovolným XMPP serverem. Správu kontaktů získaných od serveru. Posílání zpráv a výměna multimediálních dat. Součástí práce bylo, seznámit se s XMPP protokolem. Vybrat vhodnou knihovnu pro realizaci jádra klienta a pokusit se jí implementovat. Dohodnout se s kolegou Tomášem Síkorou na komunikaci mezi uživatelským rozhraním a jádrem klienta. Vymyslet strategii, která bude použita při přenosu multimediálních dat a pokusit se jí realizovat. Vytvořit testovací program pro ověření implementovaných funkcí.

Výsledné jádro XMPP klienta podporuje standardy XMPP Core[4] a XMPP IM[5]. Je schopen se spojit s libovolným XMPP serverem podporujícím tyto standardy. Jádro klienta je plně propojeno s uživatelský rozhraním a samostatně funguje pouze v testovacím programu. Strategie pro přenos

4 XMPP Core – standard pro spojení v síti XMPP.
5 XMPP IM – standard pro výměnu zpráv v síti XMPP.

multimediálních dat byla vytvořena. Tato strategie se zakládá na využití real time protokolu (RTP[6]), ale nakonec se nepodařilo jí implementovat. Jádro klienta bylo testováno na několika veřejných XMPP serverech. Nejdůkladnější testování probíhalo se serverem kolegy Zbyňka Složila.

Tomáš Sikora – Grafické uživatelské rozhraní pro jádro XMPP klienta [2]

Tato práce je, především úzce spjata s projektem Zdeňka Parmy. Mám za úkol poskytovat grafický výstup aplikace a využít funkcí jeho jádra klienta. GUI[7] klienta je řešeno pomocí jazyka QT[8]. Ačkoliv jsem uvažoval o hned několika GUI knihovnách a různých programových řešeních, QT se hodilo pro tuto práci zdaleka nejvíce.
Řešení je multiplatformní (s nutností rekompilace a linkování příslušné knihovny, jak pro operační systém Microsoft Windows, tak pro operační systém Linux) a je zde verze běžící v konzoli.
Jako contactlist jsem implementoval komponentu používanou komunikačním klientem PSI a využívám, některých jeho funkcí.
Díky vlastnostem jazyka QT nebyl problém splnit zadání, ale bohužel jsem neimplementoval multimediální funkce a nedodělal konzolové rozhraní tak, jak jsem si původně představoval.
Klient je připojitelný na jakýkoli server již podporující protokol XMPP (alespoň teoreticky, povedlo se mi připojit na všechny testované).
Stylování klienta je řešeno pomocí přiloženého .qss souboru a editace probíhá na základě vlastností specifikovaných tvůrcem jazyka.
Snažil jsem se, udělat co nejjednodušší, co nejpřehlednější a zároveň pokud možno moderní, a pro komunikačního klienta standardně vypadající rozhraní.

6 RTP (Real-Time Protokol) – protokol popisující výměnu multimediálních dat.
7 GUI (Grafic Usert Interface) - grafické rozhraní sloužící ke zobrazení výstupu, pro uživatele v přehledné formě
8 QT – knihovna pro tvorbu GUI.

2. Protokol XMPP

2.1 XML (Extensible Markup Language)

XML je obecný značkovací jazyk. Je to jednoduchý, flexibilní textový formát, vyvinutý z mnohem komplikovanějšího jazyka SGML[9]. XML bylo vyvinuto W3C[10] konsorciem. XML je strukturovaný dokument. To znamená, že všechny informace uložené v tomto dokumentu jsou strukturována do stromu. Uzly tohoto stromu tvoří značky (tag). Značky mohou být párové, počáteční značka <tag> a ukončující značka </tag>, vše mezi těmito značkami je obsah značky tag. Značky bez obsahu nemusejí být párové </tag>. Jednotlivé značky mohou i obsahovat atributy. Značka i s obsahem se nazývá element. XML neobsahuje žádné předdefinované značky. Značky je však možné definovat pomocí DTD (document type definition). Pokud je DTD nadefinováno, je možné provádět kontrolu struktury dokumentu a získávat z ní údaje pomocí programu, který se nazývá parser. Existují dva základní typy parserů. SAX (Simple API for XML) zpracovává dokument sekvenčně po jednotlivých elementech, které načítá. DOM (Document Object Model) načte celý dokument do paměti jako strukturu instancí.

XML hraje důležitou roli ve výměně dat prostřednictvím internetových služeb, jako je například právě protokol XMPP.

2.2 Stručné seznámení s protokolem XMPP [5]

XMPP (Extensible Messaging and Presence Protocol) je otevřený protokol založený na XML. Původně byl navržen pouze pro výměnu textových zpráv (instant messenging). Byl vyvinut Jabber open-source komunitou v letech 1998-1999, proto se také dříve říkalo protokolu Jabber. V roce 2000 byl termín Jabber označen jako ochranná známka. Poté, co se roku 2002 podařilo jádro protokolu prosadit do IETF[11] byl Jabber přejmenován na XMPP.V dnešní době je mezi slovem Jabber a XMPP, stejné spojení jako mezi HTTP[12] a webem. XMPP se neustále rozšiřuje o nové standarty.

V současnosti jsou v IETF vydány tyto standardy pro XMPP.

- XMPP Core [RFC 3920] – Základní vlastnosti protokolu pro navázání spojení, umožňující, co možná nejrychlejší výměnu XML elementů. Definuje základní elementy protokolu. Obsahuje také zabezpečení pomocí šifrování TLS a autentizací SASL. [6]
- XMPP IM [RFC 3921] – Přidává podporu zpráv do XMPP Core. Přesněji přidává definici elementů zprávy, presence a seznamu kontaktů. Popisuje způsoby rozesílání presencí, registraci kontaktů do kontakt listu. [7]
- XMPP CPIM [RFC 3922] – Se převážně používá pro propojení XMPP s jiným protokolem prostřednictvím brány.
- XMPP E2E [RFC 3923] – Podpora pro přímou (end-to-end) šifrovanou komunikaci.

9 SGML (Standard Generalized Markup Language) – značkovací jazyk.
10 W3C (The World Wide Web Consortium) – organizace spravující mezinárodní standardy pro www stránky.
11 IETF (Internet Engineering Task Force) – Skupina spravující standardy v síti Internet.
12 HTTP (Hyper Transport Protocol) – protokol používaný pro webové stránky.

- XMPP URN [RFC 4854] – Díky rozšiřitelnosti protokolu a vzniku velkého množství doplňků bylo založeno XMPP Standards Foundation (XSF), které se snaží obsáhnout tato rozšíření. Pro jejich používání musel být zaveden Uniform Resource Name (URN). Ten vytvořil jednotnou formu pro pojmenování veškerých rozšíření uchovávaných v XSF.
- XMPP ENUM [RFC 4979] – Mapování XMPP pomocí DNS.
- XMPP URI [RFC 5122] – Specifikuje jak interpretovat XMPP adresu jako URI.

2.2.1 Výhody XMPP [3] [4]

- je bezpečný – podporuje šifrování TLS a autentifikaci SASL. Je odolný vůči spoustě forem malweru. Je rozdělený do spousty implementací a nemá centrální prvek, který by při výpadku mohl poškodit celou infrastrukturu.
- Je decentralizovaný – je založený na client-server architektuře s velkým množstvím serverů. Decentralizace u XMPP je na stejné úrovni jako u elektronické pošty. Pokud vypadne server, zasáhne to jen malou část uživatelů. Navíc kdokoliv si může rozeběhnout vlastní XMPP server a spojit se přes síť k ostatním.
- Je rozšiřitelný – díky pružnosti XML, na kterém je XMPP založeno je možné vytvořit spousty nových možností. Dnes se dá XMPP použít v širokém množství aplikací jako hry, sociální sítě, VoIP[13], spolupráce v reálném čase, upozornění, synchronizaci dat, geolokaci, komunikaci mezi stroji, dokonce i transporty do jiných im sítí. Rozšíření se definují jako XSF
- Je to standard – jádro XMPP je definováno v IETF a rozšíření jsou definovány jako otevřené standardy spravované XSF.
- Je otevřený – otevřené standardy a velké množství softwarových produktů. Kolem XMPP se pohybuje velké množství techniků, vývojářů, open source[14] projektů a komerčních projektů. Celá tato obrovská komunita pracuje společně na vytváření spousty nových možností XMPP protokolu.
- Je prověřený – XMPP je vyvíjen již přes deset let, výsledkem je stabilní, velmi rozšířený, dobře otestovaný protokol s milióny koncových uživatelů.

2.2.2 Nevýhody XMPP [3] [4]

- Nemá heartbeat – heartbeat je funkce, která zajišťuje pravidelnou kontrolu zda-li je kontakt stále připojený, v XMPP se změny stavu posílají asynchronně. Pokud se tedy kontakt odpojí musí o tom oznámit serveru a ten rozeslat všem ostatním, že se kontakt odpojil. Při neplánovaném vypadnutí kontaktu (vypnutí proudu, pád aplikace, přerušení síťového spojení) se však žádná zpráva o změně stavu poslat nemůže a ostatní se tak stále domnívají, že je kontakt připojený a teprve po odeslání zprávy a návratu chyby se dozví, že kontakt již není připojen.
- Je rozšiřitelný – tento fakt je výhoda i nevýhoda zároveň, existuje celá řada problematických rozšíření. Díky schvalování otevřených standardů toto ovšem není zas takový problém.

13 VoIP (Voice over Internet Protocol) – protokol pro audio komunikaci.
14 Open source – otevřený, zcela dostupný kód.

- Nelze převést JID mezi servery – JID je identifikační jméno pro přihlášení na server a obsahuje v sobě i název serveru, například zdenek@jabber.cz, kde jabber.cz je server na kterém je JID registrován. Není možné se s takovým JID přihlásit například na server jabber.org, ale psát kontaktům na serveru jabber.org je samozřejmě možné. Je to podobná situace jako například u emailu.
- Chybí dominantní aplikace - Prozatím neexistuje žádný klient, který by vynikal něčím, co by přimělo velké množství uživatelů přejít z konkurenčních sítí jako je například ICQ nebo WLM. Tato nevýhoda v současné době platí už pouze pro desktopové aplikace, jelikož s nástupem facebook komunikace založené na XMPP protokolu se XMPP stala sítí pro im komunikaci s největším počtem uživatelů.
- Není binární – to nemusí být nevyhnutelně nevýhoda, ale tento protokol byl založený jako textový. Mnozí však poukazují na to, že binární přenos dat je lepší. Co se týče komunikace existují metody komprese XML zpráv. Problém však nastává při snaze přenášet data, popř. Uskutečňovat audio nebo video hovory. Proto se při implementaci posílání souborů nebo audio, video spojení používají jiná než XMPP spojení.

2.3 Srovnání XMPP s jinými protokoly

2.3.1 Microsoft notification protocol

Je protokol společnosti Microsoft využívaný pro instantní komunikaci. Je vytvořený v .NET a komunikuje přes TCP[15]. Tento protokol využívá velké množství klientů, největší sílu má však protokol s oficiálním klientem Windows live messenger (dříve známý jako MSN). Ten je součástí balíku Windows live, který nabízí velké množství služeb, které mohou spolupracovat s tímto klientem. Oproti klientům založeným na XMPP umí sdílet soubory, vytvářet audio i video hovory, podporuje hry, lze synchronizovat s poštovním klientem Microsoft Outlook. Díky tomu je také ve světě velmi populární a má již přes 300 miliónů uživatelů. Jeho nevýhodou je, že je komerční, podléhá tedy licenčním podmínkám, které nemusí všem vyhovovat. Je také silně spojený s produkty Microsoft.

2.3.2 ICQ (I Seek You)

Protokol vyvinutý v Izraeli v roce 1996. V začátcích vydařený protokol, kterému se povedlo rozšířit v Rusku, Izraeli a také u nás v České republice, snažil se i prosadit v USA, kde se neujal díky dominanci Windows live messenger. Tento protokol je komerční a spadá pod velmi přísné licenční podmínky, jako například zákaz užívání ICQ sítě osobám mladším 13 let, nutnost používat jen oficiálního klienta se zobrazováním reklamy. Tyto podmínky evidentně nejsou nijak silně vymáhány, jelikož značná část uživatelů je porušuje. V současnosti ICQ zaspal dobu a poslední verze klienta pro tento protokol, přinesla zajímavou novinku v podobě komunikace přes facebook, což je XMPP protokol. Tudíž možná časem nastane doba, kdy ICQ nenápadně přejde na XMPP.

15 TCP (Transmission Control Protocol) – Spojení s potvrzováním doručení.

2.4 Termíny [5]

I když většinu detailů z protokolu XMPP řeší knihovna, je i tak potřeba vypsat některé pojmy. XMPP je založeno na XML. Celá komunikace je vlastně jeden velký XML dokument. Části dokumentu, které se posílají jsou párové XML značky (tags). Existují tři základní značky, kterým se říká strofy (stanzas) <message/>, <presence/> a <iq/>. Jednotlivé strofy a značky zde budou obsahovat i příklad jak vypadá jejich XML část i když to není nutné znát pro používání knihoven pro práci s XMPP, je to názorná ukázka jak je XMPP protokol jednoduchý.

2.4.1 JID [RFC 3920]

JID nebo také Jabber ID, je identifikátor (adresa) entity v síti XMPP, toto id se získává většinou pomocí DNS. Každé JID obsahuje část s doménou, ta označuje server, ke kterému se daná entita připojuje. Další částí je jméno entity. JID vypadá jako emailová adresa, na některých serverech může dokonce být úplně stejná jako emailová adresa, například to lze vidět na serverech googlu, kde adresa gmailu vypadá stejně jako JID do google XMPP služby google talk. JID s jménem uživatele a serverem se nazývá „nahé“ (bare). Další částí JID je zdroj (resource), při připojení na XMPP server se přidává k nahému JID ještě další informace, ta by měla specifikovat nějakým způsobem dané přípojení. Například místo, ze kterého se píše nebo stroj, ze kterého se entita připojila. Důvod pro přidání zdroje je prostý, v XMPP protokolu je možné, aby se jedna entita mohla přihlásit vícekrát, aby to bylo možné je potřeba jednotlivé přihlášení rozlišit. Zdroj se přidává na konec adresy a takový to JID nazýváme plný (full). Příklad plného JID: petr@xmpp.org/home, kde petr je jméno entity, xmpp.org je server a home je zdroj.

2.4.2 Seznam kontaktů (Roster) [RFC 3921]

Jak už název napovídá, bude tento seznam obsahovat jednotlivé JID, které má entita registrované/podepsané. Obvykle se k nim vypisují i informace o stavu jednotlivých JID (presence).

2.4.3 Zpráva (message) [RFC 3921]

Je základní strofa pro předání informace od jedné entity k druhé. V jádru XMPP protokolu není implementace pro kontrolu doručení zprávy, z důvodu co nejrychlejšího odeslání zprávy. Zprávy se používají v IM klientech, varováních, notifikacích a podobných aplikacích.

Typy zpráv:

- normal – Tento druh zprávy je velmi podobný emailové zprávě.
- chat – Zpráva typu chat slouží pro co nejrychlejší (real-time) výměnu zpráv mezi dvěmi entitami.
- groupchat – Podobný typ jako chat, slouží, ale pro zobrazování více entitám najednou v tzv. multi-user chat místnostech.
- headline – Slouží pro varování a upozornění, u tohoto typu zprávy se neočekává žádná odpověď.
- error – Pokud nastane chyba s předchozí odeslanou zprávou, enttita která tento problem

zaregistruje vrátí zprávu typu error.

Příklad XMPP zprávy:

```
<message from="petr@xmpp.org/klient"
         to="jana@xmpp.org"
         type="chat">
    <body>Kde jsi?</body>
    <subject>Hledám</subject>
</message>
```

Na příkladu lze vidět, že je typu chat, dále obsahuje dvě značky specifické pro tento typ zprávy a to <body/>, kde se nachází text zprávy a <subject/> neboli předmět zprávy.

2.4.4 Chat State Notification [XEP-0085] [3]

Je rozšíření pro zprávu, které doplňuje stav konverzace. Jak lze vidět na obrázku 1.1 pokud jedna entita započne konverzaci, odešle druhé entitě stav Starting. Poté se stav konverzace pohybuje různě mezi Active, Inactive, Paused a Composing až do doby ukončení konverzace, tedy zaslání stavu Gone.

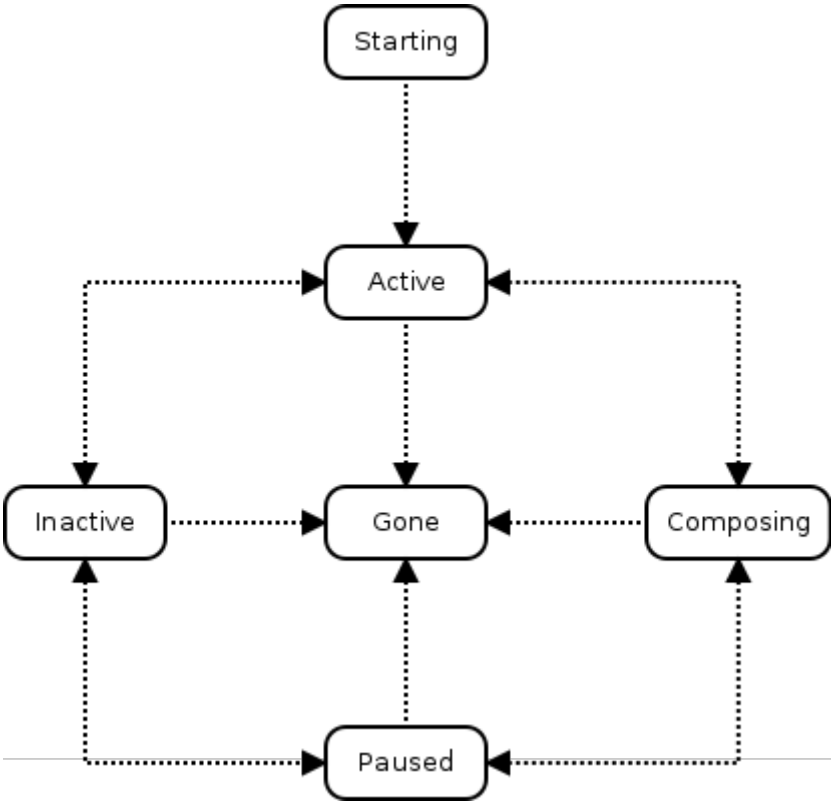
Podrobnější popis stavů:

- Starting – někdo právě začal konverzaci, které se ještě neúčastníte.
- Active – jste účastníkem konverzace, ale momentálně nepíšete.
- Composing – právě píšete zprávu.
- Paused – psali jste zprávu, ale přestali jste.
- Inactive – nevěnujete konverzaci pozornost.
- Gone – ukončili jste konverzaci (zavřeli okno).

Příklad XMPP zprávy s notifikací:

```
<message from="petr@xmpp.org/doma"
         to="jana@xmpp.org/práce"
         type="chat">
    <composing xmlns="http://jabber.org/protocol/chatstates"/>
</message>
```

Na příkladu lze vidět nová značka oznamující, že uživatel právě píše zprávu. Xmlns nám určuje jmenný prostor pro chat state notification rozšíření.



Ilustrace 1: Obr 1.1: Změna stavů notifikace.

2.4.5 Presence [RFC 3921]

Tato strofa je další důležitou součástí XMPP. Jejím účelem je informovat o dostupnosti entity, zda je připravena pro výměnu zpráv nebo není vůbec připojená apod. Tyto presence se rozesílají skupině povolených entit. Pro povolení posílání presencí je nutné provést mezi dvěma entitami podepsání (subscription) presence. Pokud se obě entity navzájem schválí, začnou se mezi nimi vyměňovat presence, kdykoliv změní stav.

Typy presence

- chat – informuje o přítomnosti a připravenosti na konverzaci.
- away – oznamuje, že je entita na krátkou dobu neaktivní. Většina im klientů ma funkci auto-away, kdy nastaví tuto presenci v době, kdy počítač nejeví žádnou aktivitu od jeho uživatele.
- xa – naznačuje delší nepřítomnost (eXtended Away). Spousty im klientů podporují i automatické přepnutí do této presence.
- dnd – pokud entita má nastavenou tuto presenci znamená to, že nemá čas a nechce být rušena (Do Not Disturb“).
- available – entita je připojena.

- unavailable – entita je odpojena.
- invalid – entita není dostupná, tato presence se objevuje například u kontaktů napojených přes transport, pokud je transport vypnutý není možné komunikovat s kontaktem ať je připojený nebo ne.
- probe – tyto presence používá server a pro implementaci klienta není důležitá.

Příklad presence:

```
<presence from="petr@xmpp.org/pda">
        <show>xa</show>
        <status>Nerušit!</status>
</presence>
```

Na příkladu je vidět, že strofa presence je velice jednoduchá, obsahuje pouze informaci od koho přišla, typ presence, tomto případě xa a zprávu, kterou entita doplňuje informace k presenci.

2.4.6 IQ (Info/Query) [RFC 3920]

Tato strofa se používá pro požadavky na server, odpovědi na události apod. Je velmi podobná protokolu HTTP a jeho požadavkům GET, POST, PUT. Na rozdíl od strofy zprávy, obsahuje IQ strofa pouze jednoho potomka, definujícího žádost nebo akci, která se má provést. IQ strofa je jediná, která musí přijmout i odpověď (od serveru nebo od entity, které se IQ strofa posílala). Odpovědi na strofy se rozeznávají podle atributu ID. IQ strofa má atribut type, který může nabývat těchto hodnot:

- get – žádost o informace, například požadavky pro registraci.
- set – poskytnutí informací entitě.
- result – odpověď na žádost o informaci nebo potvrzení o úspěšném provedení set operace.
- error – chybové zprávy.

Příklad IQ zprávy:

```
<iq from="petr@xmpp.org/pda"
                id="rr82a1z7"
                to="petr@xmpp.org"
                type="get">
        <query xmlns="jabber:iq:roster"/>
</iq>
```

Zde v příkladu žádá (type="get") petr svůj server xmpp.org o seznam svých kontaktů (xmlns="jabber:iq:roster").

```
<iq from="petr@xmpp.org"
        id="rr82a1z7"
        to="petr@xmpp.org/pda"
        type="result">
    <query xmlns="jabber:iq:roster">
        <item jid="jana@xmpp.org"/>
        <item jid="pavel@xmpp.org"/>
    </query>
</iq>
```

Jako odpověď (type="result") získal dva kontakty jana@xmpp.org a pavel@xmpp.org. Odpověď se pozná podle stejného id (rr82a1z7).

IQ zprávy jsou v této práci plně v režii knihovny. Jsou to však stěžejní strofy jádra XMPP protokolu a proto jsou alespoň okrajově zmíněny.

2.4.7 Service discovery [XEP-0030] [4]

Umožňuje připojit další služby jako místnosti pro konverzaci MUC[16] , transporty do jiných sítí a podobné služby.

2.5 Knihovny XMPP

Výběr knihovny byl pro tuto práci nejdůležitější. Při výběru byly brány v úvahu tyto kritéria:

- Objektově orientovaný programovací jazyk,
- Programovací jazyk, který již ovládám.
- Multiplatformnost – možnost používat knihovnu ve více operačních systémech.
- Dobrá dokumentace.

Podle těchto kritérií byly vybrány programovací jazyky C++, JAVA, popř. C, pokud by šlo začlenit do C++.

16 MUC (multi-user chat rooms) – místnost, kde může komunikovat více klientů najednou.

2.5.1 Knihovny programovacího jazyka C

Iksemel [8]

Knihovna vyvíjená od roku 2008. Malá a velmi modulární knihovna zaměřená hlavně na parsování XML dokumentů a hlavně XMPP formy XML dokumentu. Je postavená na POSIX prostředí a podporuje i Windows, pomocí MINGW, což je gcc kompilátor pro prostředí Windows. Podporuje SAX, DOM a XMPP parsery. Obsahuje TLS i SASL podporu. Má velmi nízké nároky na paměť. Tato knihovna je vydávaná pod licencí GNU LGPL, což umožňuje v podstatě její libovolné použití.

Její nevýhodou je, že obsahuje pouze XMPP Core protokol, tudíž jí chybí podpora pro seznam kontaktů, rozesílání presencí a všech funkcí znázorněných v XMPP IM. Práce s touto knihovnou by vyžadovala znalost přesného tvaru XML strof, což by nebyl problém, ale existují knihovny obsahující mnohem širší implementaci XMPP, tak není důvod se omezovat. Další nevýhodou je složitější propojení této knihovny a C++, u složitějších funkcí jako je připojení apod. Pro použití jako čistě parserovací knihovna je však ideální, stejně tak jako pro vytváření vlastních XML značek. Proto jí také používá kolega pracující na XMPP serveru Zbyněk Složil.

Tato knihovna se používá v projektech:

- Pardus – Správce balíčků pro PISI
- Asterisk – VoIP pro google talk
- Aubit 4GL Project – kompilátor pro BRDO aplikace

Strophe [9]

Další jednoduchá knihovna, napsaná v programovacích jazycích C a v javascriptu Už podle implementace i v javascriptu lze vytušit, že se tato knihovna zaměřuje spíše na webové klienty. Stejně jako iksemel obsahuje tato knihovna pouze XMPP Core, to znamená, že obsahuje šifrování i autentizační mechanismy. Ale kvůli absenci XMPP IM implementace, je nutné všechny zprávy, presence i správu kontaktů zpracovávat parserem a ručně vytvářet odpovědi. Strophe je vydáván pod licencí GNU GPL.

Na strophe, konkrétně na javascriptové verzi strophe, je postavený TrophyIm – XMPP klient, který je možno integrovat do webového prohlížeče.

Loudmouth [10]

knihovna zaměřená na co nejjednodušší použití, podporuje operační systémy Linux, Windows a Mac OS X. Bohužel dokumentace podle všeho neexistuje.

2.5.2 Knihovny programovacího jazyka C++

Iris [11]

Velmi slibně vypadající knihovna. Založená na Qt/C++. Iris je úzce spjata s projektem Psi (multiplatformní XMPP klient). Je neustále ve vývoji a již teď obsahuje velké množství XMPP standardů i rozšíření. Obsahuje XMPP Core i XMPP IM, tudíž není potřeba sepisovat vlastní XMPP strofy, všechno je v režii knihovny. Z rozšíření obsahuje datové formy, XMPP vyhledávání, místnosti pro konverzace, vcard, status neviditelnosti, dokonce i přenos souborů a mnoho dalších. Další skvělou vlastností Iris je jeho licence, tedy GNU LGPL. Další výhodou je navázanost této knihovny na QT, ve kterém dělá uživatelské rozhraní kolega Tomáš Sikora. Jedinou nevýhodu je slabší dokumentace. Po dlouhém rozhodovaní se však ani tato knihovna nepoužila pro tuto práci.

Projekty používající Iris

1. Psi – multiplatformní XMPP klient
2. Kopete – multiprotokolový klient integrovaný do prostředí KDE v Linuxu.

Gloox [12]

Na funkce nejbohatší knihovna ze všech, které jsou v této práci popisovány. Je napsaná v čistém ANSI C++. Je vyvíjena již třetím rokem. Oproti většině ostatních knihoven má nevýhodu, že vychází pod licencí GNU GPL. Naproti tomu, má tato knihovna výbornou dokumentaci. Je podporovaná na velké škále platforem: Linux, Windows, Mac OS X, Symbian, FreeBSD, NetBSD a OpenBSD. Tato knihovna je napsaná v čistém C++, to znamená, že jí lze bezproblémově integrovat do jakéhokoliv grafického prostředí. Vzhledem k tomu, že můj kolega Tomáš Sikora, má implementovat aplikaci nejen v Qt, ale i její konzolovou verzi, bylo čisté C++ ideální řešení. Navíc testovací program pro tuto práci je také konzolový. Právě tato knihovna byla vybrána pro tuto práci, díky její implementaci v čistém C++, její skvělé a bohaté dokumentaci a velkému množství implementovaných funkcí.

Příklady projektů využívající knihovnu gloox:

1. Jabbim server – přesněji část pro transporty.
2. qutIM – multiprotokolový im klient.
3. FatRat – manažér pro stahování z Internetu.
4. Juick – síť pro posílání krátkých zpráv.
5. Spectrum – XMPP brána do jiných im sítí.

Qxmpp [13]

Je začínající ale velmi nadějně vypadající knihovna, založená na Qt/C++. Již obsahuje jak XMPP Core, tak i XMPP IM. Je vydávána pod licencí GNU LGPL.

2.5.3 Knihovny programovacího jazyka JAVA

Emite [14]

Tato knihovna implementuje XMPP komunikaci prostřednictvím BOSH [XEP-0124]. BOSH (Bidirectional-streams over synchronous http) simuluje dlouhodobé TCP spojení mezi dvěma entitami prostřednictvím dotazů http. Obsahuje jak XMPP Core tak i XMPP IM. Je napsána v čistém jazyce JAVA.

Jabber stream object (JSO) [15]

JSO implementuje funkce pro klienta, komponenty a servery. Poskytuje nízkoúrovňovou podporu pro XMPP protokol. Z protokolu je v knihovně implementováno XMPP Core a XMPP IM. Cílem knihovny je vytvořit velmi rozšiřitelnou a lehce upravitelnou platformu pro vytváření aplikací do sítě XMPP. JSO je vydávaná pod licencí LGPL.

Smack [16]

Velice populární knihovna, poskytuje podporu jak pro vysokoúrovňový tak i nízkoúrovňový přístup k XMPP protokolu. Knihovna Smack je velmi dobře dokumentována formou prostřednictvím javadoc[17], ale i na webových stránkách výrobce. Součástí knihovny je i výkonný grafický nástroj pro vyhledávání chyb (debugger).

2.6 Knihovna gloox podrobněji

Knihovna gloox je postavena podle návrhového vzoru pozorovatel (observer). To znamená, že všechny operace knihovny jsou vykonávány pomocí událostí. Proto také nejdůležitějším nástrojem jsou obsluhy událostí (event handlers). Existují 4 typy obsluhy událostí pro základní protokol definovaný v RFC a mnoho dalších pro rozšíření XEP. Všechny obsluhy událostí použité v této práci budou v textu popsány.

Do XMPP sítě se lze pomocí knihovny gloox připojit, buď jako klient nebo jako komponenta. Komponenta se používá u serveru, například pro transporty mezi různými im sítěmi. Tato práce se zaměřuje na klienta, proto zde použití knihovny jako komponenty nebude dále rozebíráno.

2.6.1 Vlastnosti knihovny gloox

Knihovna gloox podporuje kompletní protokol XMPP Core, proto obsahuje šifrování TLS[18] i autentifikační mechanismy SASL[19] [RFC 2222]:

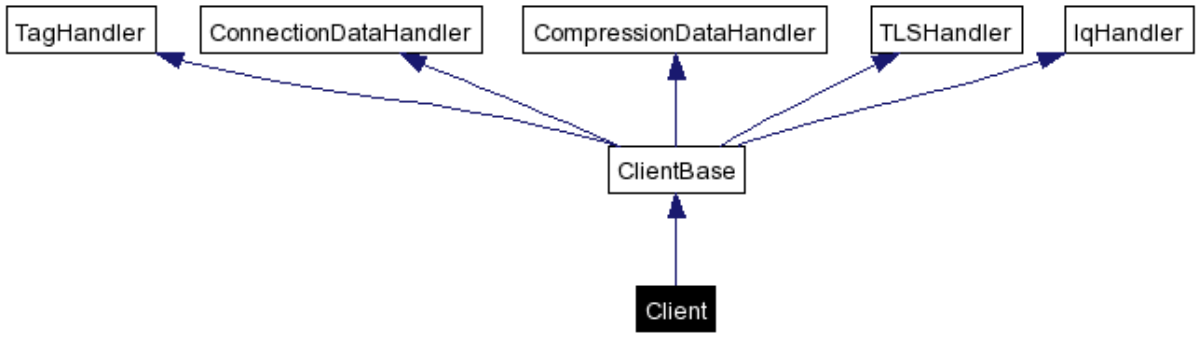
- DIGEST-MD5 – pokud server podporuje tento mechanismus, bude automaticky vybrán, jedná se o nejlepší variantu podporovanou knihovnou gloox. Jméno a heslo jsou dobře chráněny i bez použití TLS šifrování.
- PLAIN – tento mechanismus posílá jméno a heslo jako čistý text, pokud není použito

17 Javadoc – dokumentace generovaná z kódu jazyka JAVA
18 TLS (Transport Layer Security) – šifrování zabraňující odposlech na síti.
19 SASL (Simple Authentication and Secure Layer) – metody pro ověřování

šifrování dají se snadno odchytit.

- ANONYMOUS – pokud není zvoleno jméno a heslo, použije se anonymní autentizace. Když jí server podporuje, vygeneruje dočasné jméno a dovolí klientovi služby určené pro anonymní přihlášení.
- EXTERNAL – tento mechanismus funguje pouze pokud má klient certifikát a privátní klíč. Server se poté pokouší zjistit klienta, například pomocí certifikátu nebo IP[20] adresy.

Třída Client



Obr 2.1: Diagram dědičnosti třídy Client.

Třída Client je jádro XMPP klienta. Představuje jednoduchého klienta, obsahujícího všechny vlastnosti definované standardem XMPP Core. Jako parametry, při inicializaci přebírá buď server nebo JID s heslem a číslem portu. Pokud při inicializaci, získá jako parametr DNS[21] jméno serveru, pokusí se o anonymní připojení. Pokud se mu předá JID a heslo, zeptá se serveru, jaký typ autentizačních mechanismů podporuje. Klient ve výchozím nastavení vždy vybere nejsilnější zabezpečení. Tedy v ideálním případě TLS šifrování a DIGEST-MD5 jako autorizační mechanismus. Samotný klient však neobsahuje, žádný mechanismus, hlídající události spojené s připojením přes TCP/IP. Proto existuje třída ConnectionListener. Zaregistrování instance této třídy do klienta nám umožní odchytávat události, jako jsou spojení, odpojení, šifrované spojení a události spojené s posíláním proudu dat (stream event). Proudy dat z knihovny gloox se v této práci používat nebudou, protože posílání souborů není obsahem této práce a na multimediální spojení se budou využívat jiné knihovny. Součástí klienta je i vlastnost disco (service discovery). Objekt této vlastnosti se vytvoří automaticky s klientem. V této práci je disco použito pouze na předávání údajů o klientovi a verzi programu.

Připojení k serveru se po nastavení potřebných parametrů provádí funkcí connect. Tato funkce má dva režimy: blokovací a neblokovací. Blokovací režim blokuje funkci tak dlouho, dokud je udržováno spojení. Neblokovací režim ukončí funkci ihned a pro příjem dat je nutné zavolat funkci recv(). Recv() přijme data ze socketu[22] a plní jim XML parser, pokud žádná data na socketu nejsou, ukončí se. V této aplikaci byl zvolen neblokovací režim, který mnohem méně zatěžoval procesor při testování. Pokud už

20 IP – adresa v síti Internet.
21 DNS (Domain Name systém) – překládá jména domén na IP adresy.
22 Socket – spojení do sítě.

je klient připojený, přijímá zprávy do XMPP parseru. Tím však schopnosti klienta končí.

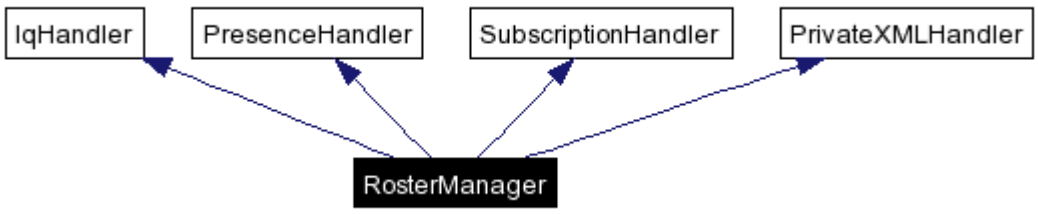
Jak už bylo zmíněno dříve, XMPP Core nepodporuje, předávání zpráv ani posílání presencí ani žádnou správu seznamu kontaktů. Pro získání funkcí z XMPP IM, je nutné registrovat do klienta další objekty. Tyto objekty zaručují obsluhu určitému druhu zpráv. To znamená, že data zpracovaná v parseru se rozešlou registrovaným objektům. Tyto objekty jsou LogHandler, RosterManager, MessageSessionHandler, MessageHandler a ChatStateHandler.

Třída LogHandler

Jak už název napovídá, tato třída nemá žádný praktický účel v aplikaci. Je však nepostradatelná pro ladění a kontrolu aplikace. Tato třída sbírá všechny data poslaná do XML parseru a vypíše je. To je obzvlášť užitečné při kontrole správného tvaru zpráv. LogHandler obsahuje i filtrování, které umožňuje filtrovat výpisy. Filtrování provádí dvěma způsoby: oblastí logování (log area) a úrovní logování. Oblastí logování je velké množství, například filtrování na zprávy z parseru, klienta, na zprávy v šifrované podobě atd. Pro ladění této operace byla ve většině případů použitá oblast LogAreaAll, neboli výpis všeho. Úrovně zpráv mají tři typy:

- Debug – zobrazí debug informace.
- Warning - varování a nekritické chyby.
- Error – výpis pouze chyb.

Třída RosterManager



Obr 2.2: Diagram dědičnosti třídy RosterManager

Pro pochopení třídy RosterManager je důležité si nejprve popsat, jak funguje seznam kontaktů (roster) v knihovně gloox. Jak již bylo zmíněné v teorii o XMPP, seznam klientů v XMPP obsahuje jednotlivé JID všech kontaktů, které si daná entita zaregistrovala. Tyto JID se rozdělují na nahé (bare) a plné (full). Plné JID má oproti nahému navíc zdroj (resource). To je část adresy, která reprezentuje stroj, klienta, místo apod. Protože je v XMPP možné se připojit pod jedním účtem (JID) vícekrát, je nutné, aby zdroj tyto spojení rozlišil. Vzniká tak zajímavá možnost, kdy máme nahé JID, které může kontaktovat více plných JID. Co se stane, když pošleme zprávu na nahé JID? Je samozřejmě možné rozeslat zprávu všem, kteří pod toto JID spadají, ale to nemusí být vždy žádaná vlastnost. Pokud by tedy klient byl například připojený neustále doma a občas se připojoval přes mobilní telefon, předpokládal by, že na mobilní telefon mu přijde zpráva vždy, zatímco na počítač doma, jen v případě, že na mobilu připojen není. To je samozřejmě možné a to díky prioritám. Každý zdroj má svou prioritu. Priorita je číslo v

rozmezí -127 až +128. Čím vyšší číslo, tím je větší pravděpodobnost, že zpráva bude doručena právě tomuto zdroji. Zpět k příkladu, pokud tedy bude mít uživatel na domácím počítači nastaveného klienta s prioritou 0 a klienta v mobilním telefonu s prioritou 10. Má jistotu, že při připojení z mobilního telefonu mu mobilní klient potlačí toho na domácím počítači.

Zde existují dva způsoby jak může klient řešit seznam kontaktů. Buď vytvoří seznam pouze z nahých JID a pokud bude chtít odeslat zprávu, pošle jí na nahý JID ze seznamu kontaktů. XMPP server přijme zprávu a na základě priorit, přepošle na správné spojení. Toto není moc korektní způsob. Za prvé tvůrci knihovny gloox to nedoporučují, protože třídy starající se o spojení mohou pracovat nekorektně z nahými JID. Za druhé, v dřívějším příkladu, pokud by chtěl uživatel po někom ze svého seznamu kontaktů, aby mu přeposlali zprávu na domácí počítač, musel by odpojit mobilního klienta.

Druhou možností je do seznamu kontaktů přidávat pouze plné JID. U každého zdroje je možné mít i jinou skupinu, tudíž i kdyby někdo z kontaktů měl příliš mnoho spojení na jednou, je možné mít stále přehledný seznam. Nebo je také možné všechny zdroje zahrnout pod jeden kontakt, tak aby uživatel ani nepoznal, že je jeho kontakt připojen z více míst a psaní na specifický zdroj mu umožnila až nějaká speciální nabídka. To už ale spíše zapadá do části grafického uživatelského rozhraní.

Nyní tedy k třídě RosterManager. Na instanci této třídy přichází z parseru všechny zprávy označené jmenným prostorem jabber:iq:roster. Slouží pro příjem JID ze seznamu kontaktů uloženého na XMPP serveru. Pokud je RosterManager zaregistrován v klientovi, oznámí o sobě okamžitě po připojení k serveru. Funkcí handleRoster vrátí celý seznam kontaktů v objektu Roster. Objekt Roster je mapa kontaktů (objekt RosterItem). Každý kontakt obsahuje jméno (name), toto jméno označuje přezdívku pod jakou si klient tento kontakt uložil. Dále obsahuje nahý JID, typ registrace (subscription). Poté obsahuje pole zdrojů a pole skupin. Skupina tu představuje textový řetězec. Zdroj je reprezentován objektem Resource, tento objekt obsahuje typ presence, status presence a prioritu.

Samotným načtením, všech plných JID ze seznamu kontaktů, schopnosti objektu RosterManager nekončí. Jak je vidět na obrázku 2.2 dědí spoustu dalších vlastností.

- IqHandler – slouží pro samotnou schopnost přijetí XML zprávy s jmenným prostorem jabber:iq:roster a protože jako všechny části gloox knihovny i tady je IqHandler využíván automaticky a není proto třeba se o něm více zmiňovat.
- PresenceHandler – poté co se načtou plné JID ze serveru, musí také dorazit jejich presence. Presence se předávají prostřednictvím funkce handlePresence, která vrací objekt Presence. Ta obsahuje typ presence, status, prioritu a pole schopností (capabilities). Schopnosti souvisejí se službami (service discovery) a předávají se v nich všechny atributy, které se nastavovaly v třídě Client pomocí objektu disco.
- SubscriptionHandler – kdykoliv někdo z kontaktů požádá o registraci, či naopak o odregistrování, instance této třídy to zachytí a informuje o tom pomocí funkce handleSubscription. Tato funkce pouze předá typ registrace a zprávu, kterou mohl doplnit k požadavku vzdálený kontakt. Třída RosterManager má samozřejmě i funkce pro požadavek o registraci.
- PrivateXMLHandler – slouží k ukládání a příjímání soukromých XML dat. Tato část

RosterManageru není v aplikaci vůbec použita, proto není třeba se o ní dále zmiňovat.

Třída MessageSessionHandler

Podobně jako u RosterManageru i zde je potřeba nejprve vysvětlit pojem MessageSession. Tato třída představuje spojení mezi dvěma entitami pro přeposílání zpráv (message). To znamená, že pro každý úplný jid (včetně resource) bude existovat jeden MessageSession. U RosterManageru bylo zmíněno, že autoři knihovny gloox nedoporučují posílat zprávy na nahý JID. To právě souvisí s MessageSession. Instance této třídy má v konstruktoru jako parametr JID a je možné jí předat nahý JID. To však může způsobovat problémy. MessageSession představuje síťové spojení. Pokud se klient připojí z různých míst, bude existovat jeden MessageSession pro dvě různá síťová spojení, což povede k nekorektnímu chování. V knihovně gloox existuje i jiný způsob implementace přeposílání zpráv, přes třídu MessageHandler. Autoři knihovny gloox doporučují používání MessageSession, které má několik výhod oproti MessageHandler:

- Instance třídy Client automaticky vytváří nové MessageSession při příchodu zprávy od JID, který ještě nemá MessageSession vytvořenou.
- Jednoduché odesílání zprávy, stačí nalézt MessageSession daného JID a zavolat funkci send. Není třeba se starat o thread_id[23], příjemce apod. Pokud MessageSession neexistuje, je nutné ji vytvořit.
- Možnost aplikovat filtry zpráv na každý jid zvlášť. Filtry zpráv přidávají další možnosti, jako různá upozornění, například oznámení, že vám druhý kontakt píše apod.

MessageSessionHandler tedy slouží pro vytváření nových MessageSession. Pokud klientu přijde zpráva od kontaktu, klient zkontroluje zda-li už má kontakt vytvořenou MessageSession, pokud ne, vytvoří ji a předá ve funkci handleMessageSession.

Třída MessageEventHandler

Odchytává jednotlivé události vyvolané kontaktem, jako odpojení uživatele, potvrzení o přijmutí zprávy, oznámení, že se zpráva kontaktu zobrazila, upozornění o psaní zprávy kontaktem, upozornění o přerušení psaní zprávy kontaktem.

MessageHandler

Tato třída se stará o získávání zpráv. Je možné jí používat dvěma způsoby. Pokud se zaregistruje MessageHandler přímo do klienta, složí jako přímý příjemce zpráv. Jak už bylo zmíněno v MessageSessionHandler, tento způsob je už autoři gloox nedoporučují. Ve spojení s MessageSession slouží MessageHandler pouze k předávání zpráv prostřednictvím funkce handleMessage. Tímto způsobem předávají jednotlivé MessageSession zprávy. Funkce handleMessage předává objekt reprezentující zprávu (message) a ukazatel na MessageSession odesilatele. Objekt Message obsahuje typ XMPP zprávy, předmět, tělo a vlákno (thread) zprávy. Typy zprávy jsou:

1. Chat – je klasická zpráva používaná v im klientech
2. Error – chybová zpráva.

23 thread_id – id, které si vyměňují dvě entity v jedné konverzaci.

3. Groupchat – skupinová zpráva, využívá se v místnostech, kde můžou psát zprávy více kontaktů najednou.
4. Headline – zpráva od které se neočekává odpověď.
5. Normal – tento typ zprávy se velmi podobá mailové zprávě.

ChatStateHandler

Je třída implementující příjem zpráv s rozšířením [XEP-0085]. Toto rozšíření definuje stavy konverzace klienta. Pomocí funkce handleChatState upozorňuje na příchozí stav konverzace a spolu s ním zasílá i JID entity posílající stav.

2.6.2 Implementace pomocí knihovny gloox

Implementace jednoduchého klienta pro instantní komunikaci byla vytvořena pomocí čtyřech základních tříd:

XmppConnect

Tato třída slouží pro připojení se do XMPP sítě a pro ladění chyb. XmppConnect je potomek třídy ConnectionListener a LogHandler. Tato třída vytváří objekt m_client, který je instancí třídy Client. Nastavuje všechny potřebné parametry: JID uživatele, heslo, zdroj, prioritu. Nastavuje také disco parametry. Mezi ně patří jméno programu a verze Součástí objektu disco je i nastavení identity a informace o tom, jaké rozšíření podporuje tento klient.

```
26: m_client->disco()->setVersion( "XmppClient", GLOOX_VERSION );
33: m_client->disco()->setIdentity( "client", "pc" );
35: m_client->disco()->addFeature( XMLNS_CHAT_STATES );
```

Připojení do XMPP sítě se provádí v neblokovacím režimu. Zde lze vidět, že klient neustále volá funkci recv(), dokud spojení nevykazuje žádnou chybu.

```
44: void XmppConnect::start() {
45:   if( m_client->connect( false ) ) {
46:     ConnectionError *ce = new ConnectionError(ConnNoError);
47:     while( *ce == ConnNoError ) {
48:       *ce = m_client->recv();
49:     }
50:       delete ce;
51:   }
51:   else {
```

```
53:     handleConnectError( "You are already connected\n" );
54:   }
55: }
```

Protože XmppConnect obsahuje nejdůležitější část klienta, objekt m_client, musí se třídám XmppRoster a XmppMessage předávat reference na XmppConnect.

XmppJid

Po seznámení se s knihovnou gloox, je jasné, že nejdůležitější částí implementace jednoduchého XMPP klienta je ukládání jednotlivých MessageSession pod správný JID. Proto byla vytvořena třída XmppJid. XmppJid reprezentuje jeden kontakt nezávisle na jeho zdroji, reprezentuje tedy nahé JID. Tato třída je potomkem třídy JID z knihovny gloox. Třída JID má možnost uložit pouze jeden zdroj, proto třída XmppJid obsahuje pole zdrojů, jejich skupin a ukazatelů na jejich MessageSession. Klient který pak jeví aktivitu se zvolí jako zdroj. Po zvolení zdroje se třída XmppJid bude jevit jako plné JID.

XmppRoster

Je potomkem třídy RosterManager. Hlavním úkolem XmppRoster je spravovat vektor uložených XmppJid.

Funkce handleRoster, která přijímá celý seznam kontaktů po připojení k serveru

```
75:  void XmppRoster::handleRoster( const Roster& roster ) {
77:     Roster::const_iterator it = roster.begin();
78:     for( ; it != roster.end(); ++it ) {
79:        XmppJid contact((*it).second->jid());
80:        contact.setNickname((*it).second->name());

88:        StringList g = (*it).second->groups();
89:        StringList::const_iterator it_g = g.begin();
90:        for( ; it_g != g.end(); ++it_g ) {
91:           contact.addGroup(*it_g);      // add group to group vector
92:           handleRosterInfo( "\tgroup: "+(*it_g)+"\n" );
93:        }
94:
95:        int resource_i = 0;              // position in vector
96:        int priority_i = -128;           // the smallest XMPP priority
97:
98:        RosterItem::ResourceMap r = (*it).second->resources();
99:        RosterItem::ResourceMap::const_iterator it_r = r.begin();

103 :        for( ; it_r != r.end(); ++it_r ) {
104:           contact.addResource((*it_r).second);   // add resource to
```

```
resource vector
    105:           handleRosterInfo( "\tresource: "+(*it_r).first+"\n" );
    106:
    107:           if( (*it_r).second->priority() > priority_i ) {
    108:              handleRosterInfo( "setResource: "+(*it_r).first+"\n" );
    109:              contact.setResource((*it_r).first);    // set priority res
    110:              contact.setGroup(resource_i);          // set priority group
    111:           }


    115:           resource_i++;
    116:        }
```

XmppMessage

Slouží pro práci s XMPP zprávami. Jak již bylo dříve popsáno, pro poslání a příjem XMPP zpráv je potřeba zaregistrovat s každým kontaktem a s každým jeho zdrojem MessageSession. Ty jsou uloženy ve vektoru XmppJid objektů. Příjem XMPP zprávy je jednoduchý a plně v režii knihovny gloox. Pokud přijde zpráva od někoho ze seznamu kontaktů (roster), knihovna gloox zkontroluje, zda-li existuje MessageSession pro JID odesilatele. Pokud neexistuje zavolá funkci handleMessageSession, ve které předá ukazatel na MessageSession. V této funkci pak musím předat tento ukazatel kontaktu do jeho instance XmppJid. Pro toto předání je třeba nejprve najít XmppJid, jelikož třída MessageSession obsahuje plné JID, stačí jej porovnat přímo s každým XmppJid (XmppJid je potomkem JID, tudíž fungují i jeho přetížené metody porovnání). Pokud najde shodu zavolá funkci newSession

```
    82:  void XmppMessage::newSession( XmppJid* to, MessageSession* session ) {
    83:     m_messageEventFilter = NULL;
    84:     m_chatStateFilter = NULL;
    85:     m_session = session;
    86:
    87:     if( m_session == NULL ) {
    88:        m_session = new MessageSession( m_client, *to );
    89:     }
    90:     m_session->registerMessageHandler( this );
    91:


    99:     to->setMsgSession(m_session);


   102:  }
```

Funkce newSession zjistí zda-li MessageSession již existuje, pokud ne vytvoří pro daný XmppJid novou. Tuto MessageSession poté musí registrovat, aby posílala zprávy na handleMessage. Dále se

nastaví filtry konverzace zpráv a nakonec se MessageSession uloží do XmppJid.

Pro odeslání zprávy musí být vytvořené MessageSession pro kontakt, kterému chceme poslat zprávu. Existují dvě možnosti jak řešit tvoření nových MessageSession. První možností je vytvořit pro každý kontakt ze seznamu kontaktů MessageSession rovnou při přihlášení k serveru. Druhou možností je prostě počkat až danou MessageSession budeme potřebovat a vytvořit jí při první výměně zprávy. Druhá metoda je výhodnější, jelikož není nutné mít v paměti neustále vytvořené všechny MessageSession.

Zde je vidět jak je odesílání maximálně zjednodušeno pomocí MessageSession. Stačí pouze najít požadované XmppJid, kterému se posílá zpráva. Zjistit jestli má vytvořenou MessageSession. Pokud ano, stačí vzít MessageSession z XmppJid a zavolat na ní funkci send a jako parametr jí předat textový řetězec se zprávou. Pokud ne, musí se vytvořit, na to se opět použije funkce newSession.

2.7 Komunikace mezi protokolem a GUI

Velkou otázkou bylo jak oddělit jádro XMPP klienta od uživatelského rozhraní. Cílem bylo, aby kolega Tomáš Sikora, nemusel znát žádné informace o tom jaké knihovny jsou použity na komunikaci přes protokol XMPP a aby nemusel ani znát žádné informace o protokolu samotném. Pro přenos informací mezi jádrem XMPP klienta a GUI se vytvořily virtuální funkce. Pro každou třídu byly vytvořeny virtuální funkce pro výpis informací a chybových hlášení.

2.8 Standard Jingle [XEP-0166]

Protokol XMPP byl původně navržen pro přenos pouze textových informací. To byl také jeden z důvodů, proč byl založen na XML zprávách. V dnešní době je však XMPP vytýkáno, že si nedokáže poradit s binárními soubory, ať už se jedná o data nebo multimédia. Proto také začaly vznikat rozšíření, která se starala o navázání spojení a přenos dat nechávala na spojení mimo XMPP síť. Jedním z takových to rozšíření je i Jingle.

Jingle je rozšíření XMPP protokolu, které má umožnit přímé (peer-to-peer) spojení mezi dvěma entitami, ať už pro přenos souborů nebo audio, popř. video spojení. Samotný jingle může definovat pouze pravidla pro spojení. Přenos dat pak bude probíhat přes spojení nesouvisejícím s XMPP. Existují však další rozšíření, která definují přenos dat v rámci XMPP sítě. Jingle je jednoduše upravitelný, aby mohl dohodnout širokou škálu typů spojení. Na příklad pro multimédia Jingle RTP session [XEP-0167]. To je modifikovaný jingle, pro umožnění RTP (Real Time Protocol) spojení, které se využívá pro přenos audia a videa v reálném čase.

3. Multimédia

Pro tuto práci byly zvažovány dvě strategie pro přenos multimediálních dat. První v podobě přenosu dat čistě v síti XMPP za pomocí Jingle. Druhá strategie využívá opět Jingle, ale tentokrát pouze pro dohodnutí počátečních parametrů spojení a o samostatný přenos dat i řízení jeho přenosu už se stará knihovna Gstreamer.

Pro popis přenosu dat výhradně pomocí Jingle je nutné navázat na předchozí kapitolu, ve které bylo popisováno XMPP rozšíření Jingle a jeho alternativa v podobě Jingle RTP session. Pro detailnější popis tohoto rozšíření je potřeba se nejprve zmínit o RTP protokolu.

3.1 RTP (Real Time Protokol) [RFC 3550] [18] [19]

RTP protokol byl vyvinut firmou Audio-Video Transport Working Group při IETF a poprvé publikován v roce 1996 jako standard [RFC 1889]. Tento standard je již dnes zastaralý a byl nahrazen novějším RFC [3550]. Jak již název firmy, která jej vytvořila vyplývá, slouží RTP pro síťové spojení umožňující okamžité (real-time) přeposlání toku dat jako je audio nebo video. RTP pro takové spojení definuje: určení užitečného zatížení (payload), číslování sekvencí, časové razítkování (time stamping) a sledování přenosu. RTP většinou běží na UDP[24] portu a dokáže vysílat jak do jednoho spojení (unicast), tak do více spojení (multicast).

RTP se skládá ze dvou částí. Z části pro přenos dat (real-time transport protokol). Ten běží na UDP portu, protože u multimediálních dat je důležitější co nejrychlejší přenos, než kontrola zda data došla v pořádku (vynechaný snímek ve videokonferenci oko nemusí ani postřehnout avšak zpozdění by zhoršovalo komunikaci). Druhou částí protokolu je řízení toku dat (real-time control protokol) RTCP , to pravidelně posílá kontrolní informace a QoS[25] (Quality of Service) parametry. RTCP běží na TCP portu, obvykle o jedničku vyšším než je RTP UDP port. TCP protokol je použit, protože kontrolní informace musí dorazit nepoškozené.

3.2 Jingle RTP session [XEP-0167]

Slouží pro započetí a výměnu přímých (peer-to-peer) spojení, pomocí protokolu RTP. Jingle RTP session musí splňovat tyto náležitosti:

- Poskytnout parametry nutné pro vznik spojení mezi dvěma entitami.
- Mapuje parametry spojení do SDP[26].
- Definuje posílání zpráv, které ovládají multimediální sezení, jako je například audio konverzace (vyzvánění, pozastavení hovoru, vypnutí zvuku).

24 UDP (*User Datagram Protocol*)

25 QoS (Quality of Service) – určuje kvalitu proudu dat (například u videa, určuje kvalitu proudu dat pro plynulé přehrávání videa).

26 SDP (Session Description Protocol) – popisuje formát parametrů zasílaných do RTCP.

Samotný Jingle RTP session nedefinuje přeposlání dat, pouze tvoří kontrolní protokol, jak je definován RTCP. Pro posílání dat prostřednictvím UDP, definuje Jingle dvě metody. Každá z nich má svůj standard. Jingle ICE-UDP Transport Method [XEP-0176] a Jingle Raw UDP Transport Method [XEP-0177].

Tato strategie přeposílání multimediálních dat má výhodu, že celá probíhá v XMPP síti. V dnešní době zatím, ale neexistuje knihovna splňující tento standard. Existuje několik knihoven, které implementují Jingle. Jsou ovšem založeny na starším standardu a nesplňují ten současný. Autoři knihovny gloox pracují na implementaci Jingle, je však teprve v raném stadiu. Vlastní implementace protokolu by byla velice rozsáhlá, proto byla vybrána druhá strategie, tedy použití Jingles pro započetí spojení a zbytek nechat na knihovně Gstreamer.

3.3 Knihovna GStreamer [17]

GStreamer je multimediální framework napsaný v jazyce C. Formou doplňků (plugin) podporuje velké množství formátů, protokolů a výstupních zařízení. GStreamer je napsaný pro co možná nejjednodušší použití. Je založen na použití tzv. rour (pipelines). Do roury pak lze připojit jednotlivé doplňky. Ty se dají rozdělit na:

- Doplňky obsluhující protokol (protocol handling).
- Zdroje (sources) pro audio a video, tyto pluginy mohou obsahovat i doplňky obsluhující protokol.
- Úpravy formátů.
- Kodeky (kodéry, dekodéry).
- Filtry
- Sinks – výstupy pro audio nebo video.

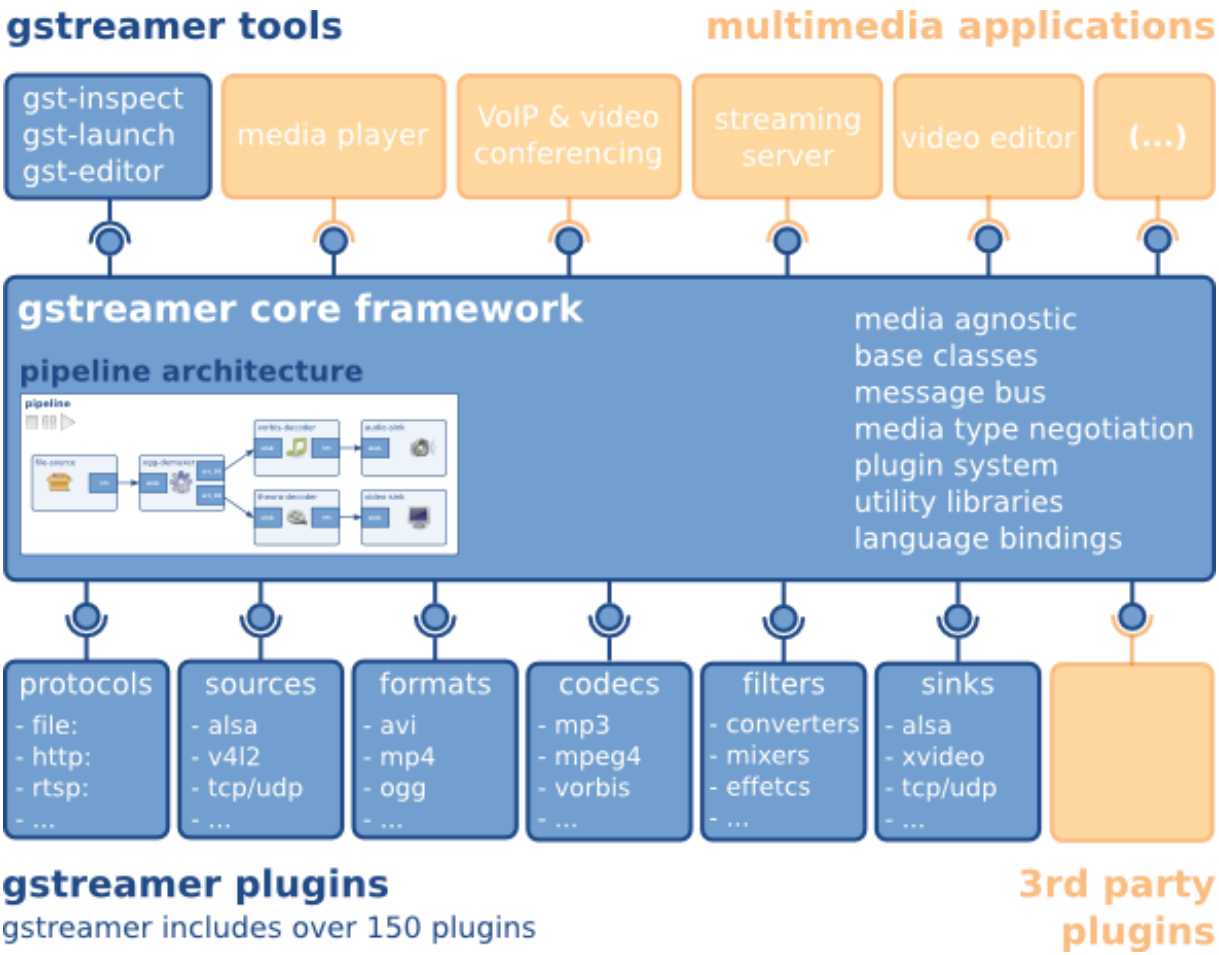
Knihovna má základní 4 prvky:

1. Elementy – jsou nejdůležitější objekty v knihovně GStreamer. Každý element má jednu specifickou funkci, jako například čtení dat ze souboru, dekódování dat, přeposlání dat na zvukový výstup apod. Právě řetězením jednotlivých elementů se vytváří funkce programu.
2. Pads – tvoří vstupy nebo výstupy elementů. Používají se pro dynamické propojení elementů.
3. Nádoby (bins) – jsou kontejnery, které obsahují elementy. Nádoba je podtřída elementu a lze s ní zacházet stejně jako s ostatními elementy. U nádoby platí, že pokud jí nastavíme nějakou vlastnost, tato vlastnost se přenese i na všechny elementy, které nádoba obsahuje.
4. Roury – jsou stejné jako nádoby. Roura je vždy nejvyšší prvek a určuje v jakém pořadí se budou spouštět elementy
5. Komunikace – představuje výměnu dat mezi aplikací a rourou.
    1. Vyrovnávací paměť (buffer) – využívá se u stahování proudu dat (od zdroje směrem k

sinku).

2. Události (events) – jsou objekty posílané mezi elementy nebo od aplikace k elementům.
3. Zprávy (messages) – jsou objekty posílané od elementů přes sběrnici aplikaci.
4. Fronty – umožňují aplikaci žádat o informace z roury.

Pro vytvoření RTP spojení budou potřeba elementy pro vstup a výstup UDP. Dekodér a kodér kodeku určenému pro posílání přes protokol RTP. Fronta udržující konstantní tok dat. Vstup a výstup zvukové karty. Nejdůležitější element bude nádoba obsluhující RTP protokol. Tato nádoba se v knihovně GStreamer nazývá GstRtpBin. GstRtpBin obsahuje vstupní a výstupní sinky pro RTP část protokolu, která se napojuje na UDP element. Také obsahuje vstupní a výstupní sinky pro RTCP část protokolu.



Obr. 3.1: přehled knihovny Gstreamer

4. Testovací program

Testovací program byl vytvořen s konzolovým rozhraním v čistém programovacím jazyce C++, pro demonstraci funkcí a testování. Program je velice jednoduchý, využívá jedno vlákno pro připojení k serveru a zbytek programu obsluhuje příkazovou řádku. Program byl používán pouze v operačním systému Linux. Program se pouští bez jakýchkoliv parametrů.

Konzolové rozhraní obsahuje tyto příkazy:

1. connect <name> <server> <resource> <password> - slouží pro připojení do sítě XMPP.
2. set – slouží k nastavení různých vlastností
    1. log <on/off> - zapne výpis nezpracovaných XML zpráv.
    2. notify
        1. typping <name/JID> - pošle kontaktu oznámení o tom, že se mu píše.
        2. receive <name/JID> - pošle potvrzení, že zpráva došla v pořádku.
        3. noactive <name/JID> - pošle oznámení, že klient není momentálně aktivní.
        4. close <name/JID> - pošle oznámení, že klient zavřel okno.
    3. Presence – nastavení vlastní presence.
        1. online
        2. away
3. show
    1. roster – zobrazí seznam kontaktů.
    2. Connect – informuje o tom, zda-li je klient připojený.
4. send <name/JID> - odešle zprávu kontaktu.
5. disconnect – odpojení z XMPP sítě.

5. Závěr

Výsledné jádro XMPP klienta splňuje mnoho ze standardu XMPP Core a XMPP IM a přidává i podporu upozornění na stav konverzace. Nejdůležitější cíl této práce byl splněn a to seznámení se z XMPP protokolem. Byly prohlédnuty rozšířenější knihovny pro XMPP klienty v jazycích C, C++ a JAVA. Každá z těchto knihoven vyhovuje jinému účelu. Byla vybrána knihovna s nejlepší dokumentací a s největším počtem implementovaných vlastností. I při použití knihoven bylo nutné proniknout do protokolu XMPP hlouběji. Nejen pro samotnou zajímavost problematiky, ale hlavně při studiu problematiky posílání multimediálních dat. Strategie pro přenos multimediálních dat je v práci popsána dvěma způsoby. Implementace těchto strategií nakonec nebyla realizována.

V této práci se vyskytuje i několik chyb. Při odpojení ze sítě se občas vyskytne chyba v zámcích[27]. Protože zámky jsou čistě v režii knihovny, může se jednat o chybu knihovny samotné. Nebyla však nahlášena žádná podobná chyba na stránkách knihovny gloox. Díky náhodnosti chyby se nepodařilo najít její příčinu. S XMPP serverem kolegy Zbyňka Složila se tato chyba nevyskytla. Další chybou je objevující se únik paměti, při přerušení internetového spojení a nasledného pokusu o odpojení ze sítě. Třídu obsluhující stavy konverzace (chat state notification) se také nepodařilo zprovoznit.

27 Zámek (mutex) – funkce systému zabraňující souběhu vláken.

Literatura

[1] Složil, Zbyněk. Bakalářská práce: Server pro protokol XMPP s podporou multimédií. Ostrava. 35 s. Bakalářská práce na Fakultě elektrotechniky a informatiky Technické Univerzity Ostravy. Vedoucí bakalářské práce Ing. David Seidl.

[2] Síkora, Tomáš. Bakalářská práce: Textové a grafické rozhraní klienta protokolu XMPP s podporou multimédií, Ostrava. 42 s. Bakalářská práce na Fakultě elektrotechniky a informatiky Technické Univerzity Ostravy. Vedoucí bakalářské práce Ing. David Seidl.

[3] Peter Saint-Andre, Kevin Smith, Remco Tronçon. XMPP: The Definitive Guide. Building Real-Time Applications with Jabber Technologies. První vyd. Sebastopol: O'REILLY, 2009. ISBN: 978-0-596-52126-4.

[4] Dana Moore, William Wright. Jabber: Developer's Handbook. Pvní vyd. Indianapolis: Developer's Library. 2004. ISBN : 0-672-32536-5.

[5] XMPP Standards Foundation. [online] URL: <http://xmpp.org/> [Cit. 2010-04-15].

[6] Extensible Messaging and Presence Protocol (XMPP): Core [online] URL: <http://www.ietf.org/rfc/rfc3920.txt> [Cit. 2010-04-15].

[7] Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence [online] URL: <http://www.ietf.org/rfc/rfc3921.txt> [Cit 2010-04-15].

[8] Iksemel [online] URL: <http://code.google.com/p/iksemel/> [Cit. 2010-04-15].

[9] Strophe [online] URL: <http://code.stanziq.com/strophe/> [Cit. 2010-04-15].

[10] Loudmouth [online] URL <http://groups.google.com/group/loudmouth-dev/> [Cit. 2010-04-15].

[11] Iris [online] URL: <http://delta.affinix.com/iris/> [Cit. 2010-04-15].

[12] Gloox. [online] URL: <http://camaya.net/gloox/> [Cit. 2010-04-15].

[13] QXmpp [online] URL: <http://code.google.com/p/qxmpp/> [Cit. 2010-04-15].

[14] Emite [online] URL: <http://code.google.com/p/emite/> [Cit. 2010-04-15].

[15] Jabber stream object [online] URL: <https://jso.dev.java.net/> [Cit. 2010-04-15].

[16] Smack [online] URL: <http://www.igniterealtime.org/projects/smack/> [Cit. 2010-04-15].

[17] Gstreamer: open source multimedia framework [online] URL: <http://www.gstreamer.net/> [Cit. 2010-04-15].

[18] RTP: A Transport Protocol for Real-Time Applications. [online] URL: <http://tools.ietf.org/html/rfc3550> [Cit. 2010-04-15].

[19] RTP: Profile for Audio and Video Conferences with Minimal Control [online] URL: <http://tools.ietf.org/html/rfc3551> [Cit. 2010-04-15].